ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA DE COMPUTADORES

MUSEUM: Una aplicación de visionado de contenidos multimedia para Smart TV

MUSEUM: A multimedia content browser app for Smart TV

Realizado por
Óscar Carmona Márquez
Tutorizado por
Daniel Garrido Márquez
Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, Diciembre 2015

Fecha defensa:
El Secretario del Tribunal

Resumen: Tradicionalmente, la televisión ha sido siempre el mejor medio (y hasta no hace mucho, el único) para visionar contenidos. Varios elementos, como áreas de visionado cada vez mayores, distintas posibilidades de conectividad y su presencia en la principal estancia del hogar lo siguen manteniendo como el dispositivo ideal para visionar contenidos.

Gracias al acceso a internet que incorporan los últimos modelos se abre un nuevo abanico de posibilidades, permitiendo el acceso (en teoría, depende de las capacidades de cada aparato) a todo el contenido disponible en la red. Esto, junto a la gran área de visionado que ofrecen, lo hacen el elemento ideal para ver contenidos de forma grupal (tanto elementos educacionales como de ocio, para la familia, amigos o cualquier grupo de personas).

Aunque cada día los fabricantes convergen hacia un estándar común que permita que una misma aplicación se ejecute en una televisión de cualquier marca sin tener que ser adaptada, lo cierto a día de hoy es que no existe una compatibilidad 100% entre aparatos, por lo que se elige para el desarrollo la plataforma más extendida hasta el momento, Smart TV de Samsung. No obstante, con pequeñas modificaciones en la aplicación se podría hacer compatible con aparatos de otras marcas.

En este trabajo fin de grado se ha desarrollado un sistema que simula una visita virtual a un museo, donde se puede acceder a los contenidos disponibles en las distintas salas en forma de vídeos, fotografías o audios a la vez que se ofrece la opción de ver una descripción en texto sobre el contenido. El contenido multimedia se almacena en un repositorio externo (servidor) que envía a la televisión dicha información bajo petición. La estructura del contenido es adaptable de forma que, con pequeños cambios, un programador puede modificar la aplicación para ajustarla a otros casos de uso.

De manera más específica:

- Un usuario con capacidades de administración en el servidor puede:
    - Añadir datos de nuevos museos
    - Organizar la estructura del museo en distintas salas.
    - Añadir contenido multimedia a cada sala.
- Un usuario que instala la aplicación en su TV puede:
    - Navegar por los distintos museos y salas definidos en la aplicación.
    - Visualizar el contenido multimedia (vídeo, fotos, audio, texto) que ofrece cada sala.

Palabras Clave: Samsung Smart TV, aplicación Smart TV, TV interactiva, TV conectada a internet, Museum, visor contenidos multimedia TV.

Abstract: Traditionally, TV has been the best device (and until not very much time ago, the only one) to browse visual content. Nowadays, a big visual surface, Internet-connection capabilities and its place in the principal room of a home keeps the TV as the best device to deploy visual content.

The access to Internet that latest TV models supply make possible to browse all the content the net offers in your TV. This, and its big visual surface, make it the perfect device to watch content for a group of people (with educational or entertainment purposes, for family, friends or any kind group of people).

It seems that TV developers are trying to accomplish a common standard, but the truth today is that a TV app is not full-compatible between different TV models. So, for this development it has been used the most extended platform, Samsung Smart TV. The developed app can be made compatible with other TV models with minimal changes.

The system developed for this project offers a virtual visit to a museum, where users can access content in different formats (video, audio, pictures, text). Content is stored in an external system (server), which sends it to the TV under petition. Content and its structure is adaptable, so a programmer can make the needed changes to adapt the app to other uses.

In a more specific way:

- A user with administrator capabilities in the server can:
  - Add new museums data.
  - Define the museum´s room structure.
  - Add multimedia content to each room.
- A user that install the app in the TV can:
  - Browse each museum and room defined in the app.
  - Browse the multimedia content of each room (video, audio, pictures, text).

Keywords: Samsung Smart TV, Smart TV app, Internet connected TV, Museum, TV multimedia content browser.

Índice de contenidos

1. Introducción

1.1 Objetivos

El objetivo de este Trabajo Fin de Grado (*TFG*) es el de desarrollar una aplicación (*app*) para *Samsung Smart TV* que simule una visita virtual a un museo. Se han recreado distintos museos reales, modelados como conjunto de salas, cada una a su vez modelada como un conjunto de obras artísticas. Estas obras se pueden presentar al usuario en distintos formatos multimedia (vídeo, audio, foto o texto). Con ello, se pretende explotar las posibilidades de presentación de distintos formatos multimedia que ofrece una TV moderna con conexión a Internet.

1.2 Estado del arte

Las aplicaciones para *Samsung Smart TV* se presentan oficialmente en un mercado ('*market*') de aplicaciones propiedad de la marca *Samsung*. Este market no es más que un repositorio desde donde los usuarios pueden descargar e instalar las distintas aplicaciones desarrolladas, tanto por la propia Samsung como por desarrolladores independientes.

Este market tiene muchas similitudes con otros mercados de aplicaciones (como *Google Play* para aplicaciones móviles *Android* o *AppStore* para dispositivos de *Apple*), aunque también tiene sus particularidades:

· Los markets están disponibles por países/zonas, por ejemplo, hay un mercado de aplicaciones para Reino Unido (UK) y otro para España. Dispondremos de acceso únicamente al market que nos corresponda según nuestra localización.

· Para poder subir una aplicación al market, hay que tener el beneplácito de *Samsung*. Para ello, hay que darse de alta como desarrollador y la aplicación debe pasar unos test de calidad antes de ser ofrecida para descarga a los usuarios.

· Cada año van surgiendo nuevas características y mejoras en los televisores fabricados. Para que estas mejoras estén a disposición de los desarrolladores de aplicaciones, *Samsung* va actualizando su software de desarrollo de aplicaciones (*SDK*, *software development kit*). Como desarrolladores, debemos tener muy presente para que versión de TV (año de desarrollo de la TV) queremos programar. Cuanto más moderna sea la versión,

más capacidades y características tendrá el dispositivo (como reconocimiento de voz o de gestos), pero el público objetivo será menor (menos unidades vendidas). Lo ideal es encontrar un compromiso entre funcionalidad y número de usuarios objetivo.

A pesar del enorme público potencial, la mayoría de aplicaciones existentes en los markets pertenecen a grandes compañías que ya ofrecían sus servicios en la web y han adaptado su contenido al formato TV. Los principales proveedores son medios audiovisuales (cadenas de televisión, periódicos, servicios de vídeo en streaming, etc.). Existen pocas aplicaciones originales desarrolladas por programadores independientes.

1.3 Estructura de la memoria

En esta memoria se describe todo el proceso de creación de una aplicación para una *Samsung Smart TV*, desde el análisis de los SDKs de *Samsung* disponibles para el desarrollo de la aplicación, la justificación del SDK finalmente seleccionado para el desarrollo, las consideraciones necesarias a tener en cuenta a la hora de desarrollar una aplicación que va a ser visionada en un TV.

También se detallan las fases de análisis y diseño de la aplicación (arquitectura necesaria) y desarrollo de código de todos los componentes necesarios.

Una vez implementada la aplicación, se explica el proceso de instalación y ejecución de la aplicación en una *Samsung Smart TV*, así como el proceso necesario para conseguir publicar una aplicación en el market de *Samsung.*

Finalmente, se concluyen con unas indicaciones para una posible ampliación o adaptación del desarrollo a otros propósitos más generales.

1.4 Tecnologías a utilizar

Para el desarrollo y prueba de este sistema es necesario contar con un PC con el software apropiado instalado, que consta de:

· Software de desarrollo de la aplicación (SDK de *Samsung*, versión 4.5, compatible con TV fabricadas hasta 2013).

Adicionalmente, si se quiere ir testeando la aplicación sin necesidad de tener que instalarla en la TV, es necesario contar con un emulador de *Samsung*

Smart TV (correspondiente al SDK usado) y un entorno que posibilite su ejecución:

· Máquina virtual donde se ejecutará el emulador de *Samsung Smart TV* (*Virtual Box* de *Oracle*).

· Archivo con extensión *.ova* conteniendo la imagen virtual del emulador, en la versión equivalente al SDK de desarrollo (*2013 Smart TV Emulator 4.5*).

	2010	**2011**	**2012**	**2013**	**2014**
Versión SDK / Emulador	1.5	2.5	3.5	4.5	5.1
HTML	HTML 4.01	HTML 5	HTML 5	HTML 5	HTML 5
DOM	DOM 2	DOM 2	DOM 3	DOM 3	DOM 3
CSS	CSS 2.1	CSS 3	CSS 3	CSS 3	CSS 3
Javascript	Javascript 1.6	Javascript 1.8	SquirrelFish	V8	JSC
SO	Linux 2.6				
Resolución	**App**	**Smart Hub**	**Video**		
	960 x 540	1280 x 720	1920 x 1080		

Tabla 1.1. Algunas características de las TVs según su año de fabricación

· Servidor web donde alojar la aplicación, con el fin de ser accedida por la TV para su instalación. No es necesario que esté alojado en Internet, con que tenga conectividad con la TV es suficiente (*Apache Server 2.2* instalado de forma local). Aunque no es necesario que sea en el mismo servidor web, éste se usará también para alojar el repositorio de los archivos multimedia que la aplicación necesita (archivos de vídeo, audio y foto).

Aunque queda fuera del alcance de este proyecto, dada la arquitectura cliente-servidor del sistema desarrollado, para una implantación real se necesitarían también los sistemas del lado del servidor:

· Aplicación del lado del servidor, que proporcione a la aplicación instalada en la *Samsung Smart TV* los datos que necesita. Es decir, proporciona una interfaz entre la aplicación y el almacén de datos.

· Sistema gestor de base de datos, donde se almacenan los datos de la aplicación en sí (museos, salas, obras, etc.), que es accedido de forma remota por la aplicación del lado del servidor.

Gráfico 1.1. Emulador de 2013 Samsung Smart TV, versión 4.5

2. Análisis de la aplicación

Para implementar la funcionalidad que se requiere se ha pensado en desarrollar un entorno basado en un esquema cliente-servidor, donde el cliente es la aplicación instalada en la TV y el servidor una interfaz (en forma de servicios web, por ejemplo) que proporciona los datos requeridos a la aplicación cliente.

Se ha considerado que esta arquitectura es la que mejor se amolda a las necesidades del desarrollo, aunque ello no implica que haya que seguirla para desarrollar una aplicación para *Samsung Smart TV*. Es posible crear una aplicación para *Samsung Smart TV* que no necesite del desarrollo de ningún componente adicional para su correcta ejecución en una TV. Pero el mantener los datos separados de la aplicación en sí hace que un cambio en los datos no conlleve una actualización de la aplicación.

De forma general, la arquitectura del sistema queda de la siguiente forma:

· Aplicación cliente. Aplicación instalada en la *Samsung Smart TV*.

· Aplicación servidor. Aplicación alojada en un servidor (en forma de servicio web, servlet, etc.) que responde a las peticiones de datos por parte del cliente.

· Almacén de datos. Base de datos conteniendo la estructura organizativa de la aplicación.

· Repositorio multimedia. Almacén de los archivos multimedia que se mostrarán en la aplicación.

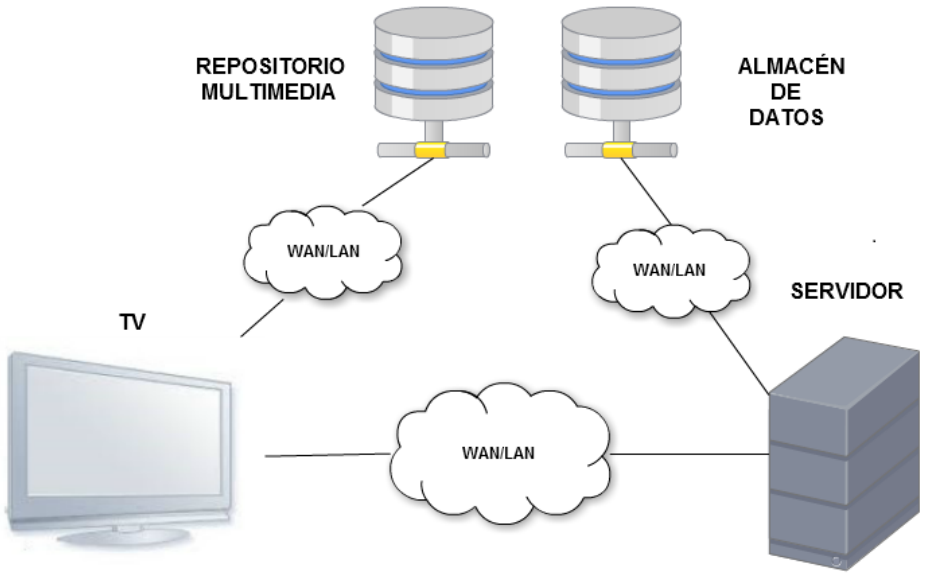

Gráfico 2.1. Arquitectura general

2.1 El lado del cliente

Ésta es sin duda la parte principal del desarrollo. Es la parte con la que el usuario interactúa directamente, por lo que debe proporcionar la funcionalidad que se espera, presentándola de una manera atractiva, intuitiva y funcional, teniendo muy presente el dispositivo donde se va a ofrecer (la TV).

Para recrear la visita virtual a un museo, se considera ésta de manera análoga a una visita real. Así, según el modelo finalmente implementado, cada museo está compuesto por un conjunto de salas, y éstas a su vez contienen un conjunto de obras, que son las que el usuario puede contemplar en última instancia. Cada una de estas obras dispone de contenido descriptivo, que puede estar disponible en distintos formatos (vídeo, audio, foto o sólo texto) o un conjunto de varios de ellos (como foto mas texto descriptivo adicional, por ejemplo). Se facilitan los mecanismos adecuados, tanto para la navegación por la jerarquía organizativa del contenido como para el visionado final de éste.

2.2 El lado del servidor

El servidor es el encargado de proporcionar al cliente los datos que éste requiera. Constituye una capa de abstracción entre el cliente y el almacén de datos, de forma que no sea necesario que el cliente acceda a este almacén de forma directa.

La misión del servidor es la de estar escuchando las posibles peticiones de datos procedentes de clientes y, cuando éstas ocurran, atenderlas. Esta atención consiste principalmente en obtener la información que el cliente requiere y proporcionársela. Cuando el cliente requiera una información concreta, la aplicación servidor realizará una petición de datos al almacén de datos para que éste se la proporcione. Una vez obtenida dicha información del almacén (o un mensaje de error, si hubiera habido alguno durante el proceso), se transforma a un formato de mensaje acordado con el cliente y procede a su envío, satisfaciendo así la petición inicial del cliente. De ésta forma, se mantiene la independencia del código a la hora de futuras ampliaciones de funcionalidades.

2.2.1 Almacenamiento de datos

Con el fin de proporcionar escalabilidad y flexibilidad, se ha optado por usar un repositorio de datos externo. Es decir, los datos a usar no van incrustados en la

aplicación cliente, sino que existe un almacén de datos que contiene la estructura organizativa de la aplicación y los datos finales que se presentan al usuario. De esta manera, se hace a la aplicación adaptable a cambios en los datos de forma que añadir un nuevo museo, por ejemplo, sea tan fácil como añadir los datos correspondientes a ese museo.

2.3 Comunicación cliente-servidor

Para la correcta ejecución de la aplicación es necesario que haya un canal de comunicación entre el cliente y servidor, dado que el cliente debe presentar unos datos al usuario que están alojados remotamente y se obtienen precisamente a través del servidor. Este canal será una red de comunicaciones (normalmente Internet) aunque también es posible usar un entorno de red local (sobre todo en entornos de pruebas). **Sin este canal de comunicación**, la aplicación funciona (se ejecuta) pero no presenta ningún dato al usuario.

Con objeto de minimizar el tráfico de datos se ha intentado minimizar las comunicaciones. Para ello, se ha evitado en la medida de lo posible redundar datos, es decir, sólo se pide un dato al servidor una vez, si no se ha pedido con anterioridad. Para conseguir esto, es necesario crear unas estructuras de almacenamiento de datos en el lado del cliente, que estudiaremos en el apartado 3.1. De esta forma, la primera vez que se recupera la información del servidor se almacena internamente en el lado del cliente y futuras peticiones de esa misma información la obtendrán del almacén local, sin generar nuevas comunicaciones ni tráfico de red.

Para este TFG se ha considerado que el lado del servidor (incluyendo el almacenamiento de datos y las comunicaciones) queda fuera de su alcance, por lo que sólo se describe su arquitectura y diseño con carácter informativo de cara a futuras ampliaciones del desarrollo.

3. Diseño de la aplicación

3.1 Cliente

A la hora del diseño del lado del cliente hay que tener muy en cuenta el medio final de presentación, la TV. Las características intrínsecas de este medio ofrecen tanto ventajas como inconvenientes frente a otros dispositivos habituales:

Ventajas:

- Gran superficie de visionado. Ideal para contenidos multimedia.
- Ubicado habitualmente en la estancia principal del hogar. Visionado normalmente por más de una persona. Ideal para contenidos de entretenimiento o didácticos.

Inconvenientes:

- Posicionado del espectador a una distancia considerable. A tener en cuenta a la hora de elegir el tamaño de los elementos a mostrar, ya que deben ser distinguibles (tamaños mínimos de fuentes, fotos, etc.).
- Interfaz de usuario pobre. Aunque existen dispositivos adicionales (como teclados, ratones, gamepads, dispositivos apuntadores, etc.), la mayoría de usuarios interaccionan con la TV a través del mando de control remoto, que proporciona una interfaz muy limitada.

Dadas estas premisas, existe una serie de recomendaciones orientadas a ofrecer una UI (User Interface, Interfaz de Usuario) optimizada al medio de presentación:

- Simplicidad. Tener una disposición de elementos clara y fluida, poco recargada. Evitar niveles de navegación innecesarios. Navegación entre niveles intuitiva. Que no sea necesario un manual para usar la aplicación.
- Claridad. Navegación entre elementos previsible, que no dé lugar a confusión. Una navegación confusa hace que el usuario no quiera seguir navegando y deje de usar la aplicación.
- Control del usuario. El método de control y los elementos representados en la pantalla deben ser acordes con el dispositivo controlador. Los botones representados deben funcionar como se esperan.
- Consistencia. Mantener la consistencia en la navegación, disposición de elementos en pantalla y controles. Proporcionar una experiencia de aprendizaje de la aplicación intuitiva y fluida.

- Feedback. Mantener al usuario informado de las consecuencias de sus acciones. Identificar claramente dónde está el foco.
- Estética. Una estética agradable hace que el usuario tenga una experiencia de uso positiva y se canse menos de usar la aplicación.

Teniendo en cuenta estas consideraciones y la naturaleza de la aplicación, las distintas pantallas que conforman la aplicación se han diseñado manteniendo una uniformidad, excepto en aquellos casos que se ha considerado que era necesario modificar el diseño estándar para mejorar la experiencia de usuario. Se decide estructurar la aplicación de forma jerárquica, en cuatro niveles: museos, salas, obras y detalle de la obra.

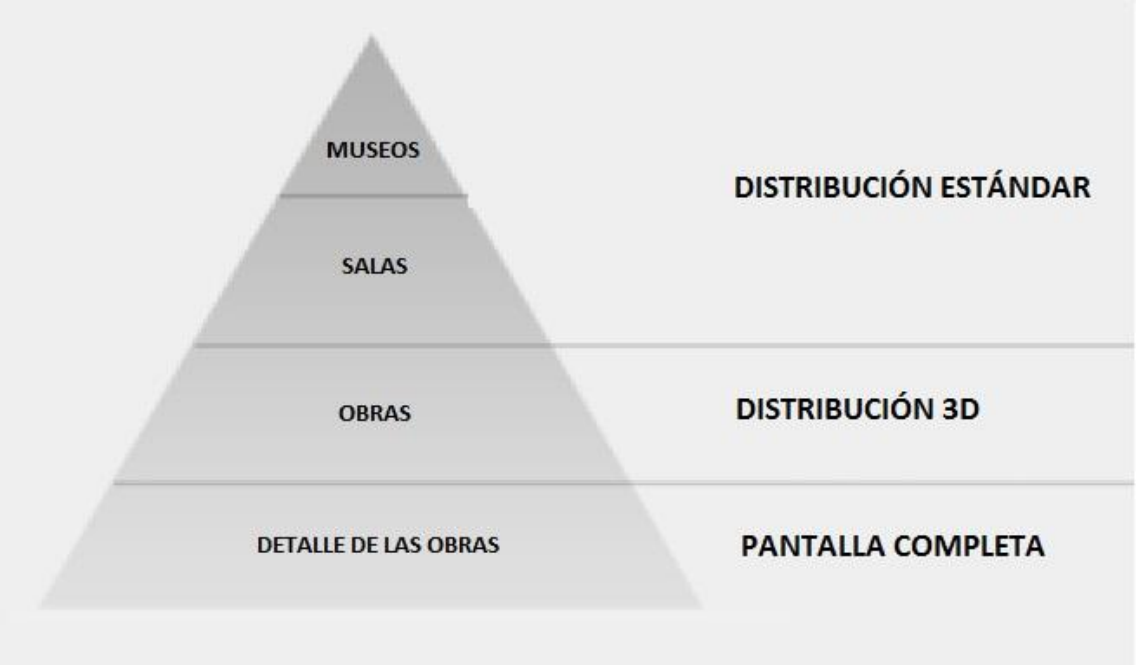

Gráfico 3.1. Diseños aplicados según el nivel de navegación dentro de la aplicación

3.1.1 Distribución estándar

Cuando se inicia la aplicación, el primer nivel que nos encontramos es el de *Museos*. Tanto en este nivel como en el siguiente (*Salas*) la pantalla de la aplicación presenta un diseño estándar. Los distintos elementos se distribuyen de la siguiente manera:

- Cabecera: Franja superior que contiene el logo (nombre) de la aplicación.

- Barra de navegación: Franja inmediatamente inferior a la cabecera, donde se va mostrando el nivel de navegación dentro de la aplicación donde se encuentra el usuario en cada momento.
- Área de contenido: Ocupa la mayor parte del área de visionado. Muestra un listado conteniendo los datos referentes al nivel donde se está (listado de museos o listado de salas de un museo). Por cada registro del listado, se muestran sus datos asociados, como el nombre, descripción, foto, etc.).
- Barra de ayuda: En la parte inferior del área de visionado se muestra una barra de ayuda que proporciona información sobre qué botones del control remoto tienen uso. Estos botones pueden variar de un nivel a otro, o según la acción que esté llevando a cabo el usuario.
- Barra de scroll: En el caso de que el listado a mostrar tenga más elementos de los que se pueden mostrar en el área de contenido, se muestra en el lado derecho una barra de scroll, indicando al usuario de que el listado dispone de más elementos de los que se están visionando actualmente. El paso de página se produce mediante una animación, que muestra el siguiente contenido disponible.

Gráfico 3.2. Distribución estándar

3.1.2 Distribución 3D

Cuando se entra al nivel de *Obras*, la distribución de la pantalla varía ligeramente. Con la intención de proporcionar un elemento más visual, acorde a lo que se está intentando modelar en la aplicación, el *área de contenido* pasa de tener un listado en modo texto a mostrar una galería en 3D conteniendo

imágenes de las obras de la sala. La obra que se encuentra en posición frontal muestra, además, su título, autor y unos iconos que indican si dispone de descripción de vídeo o audio. Navegando por las distintas obras se va mostrando una animación que hace el efecto visual de rotación.

Se puede apreciar que la *barra de ayuda*, aunque sigue presente, ha cambiado los botones que muestra, actualizando la funcionalidad de la aplicación.

Gráfico 3.3. Galería 3D para el visionado de las obras de una sala

3.1.3 Pantalla completa

Si se selecciona una obra de la galería, se entra a su visionado en detalle, a pantalla completa. En este modo, la idea es usar toda la superficie de visionado posible para mostrar la obra en sí (una imagen de la obra), por lo que desaparecen elementos de diseño como la *cabecera* y *la barra de navegación*. La *barra de ayuda* se mantiene.

Para resolver la muestra de datos adicionales (video, audio o texto descriptivo) sin entorpecer la visión de la obra se ha optado por mostrar estos elementos en capas superpuestas sobre la imagen de la obra. Pulsando distintos botones, se hacen visibles mediante una animación la capa del texto descriptivo o la de vídeo (o un mensaje de error, si no lo hubiera). El audio, al no ser un elemento visual, se reproduce al pulsar el botón correspondiente sin necesidad de mostrar ninguna capa adicional. Esta capa sólo se haría visible en caso de que

fuera necesario mostrar un mensaje de error (no existiera archivo de audio relacionado).

En los casos en los que se dispone de archivo multimedia (vídeo o audio) y éste empieza a reproducirse, la *barra de ayuda* se actualiza con valores de los botones de control de reproducción.

Con el fin de mejorar la navegabilidad, el usuario puede ir visionando las siguientes obras sin necesidad de volver al nivel de la galería, rotar hasta la siguiente obra y seleccionarla. Si realiza esta acción mientras se está mostrando la capa del texto descriptivo, ésta permanece visible y se actualiza con la información de la nueva obra seleccionada. Esta persistencia del texto descriptivo se mantiene incluso si se vuelve al nivel de galería y se selecciona otra obra. En cambio, si se estaba reproduciendo vídeo o audio, éste se detendrá y su capa de visualización correspondiente se ocultará.

Gráfico 3.4. Pantalla completa, mostrando la capa flotante de texto descriptivo

3.2 Servidor

En esta sección se va a describir la definición conceptual de la arquitectura necesaria del lado del servidor, haciendo constar que no se llega a implementar dicha arquitectura por quedar fuera del alcance de este TFG.

3.2.1 Modelo de datos

La información que necesita la aplicación es simple y está bien estructurada de forma jerárquica. Consta de un conjunto de **museos** *(museums)*. Cada museo está formado por un conjunto de **salas** *(rooms)*, y cada sala está formada por un conjunto de **obras** *(artworks)*. Cada uno de estos tres elementos tendrá sus atributos correspondientes. En detalle, serían:

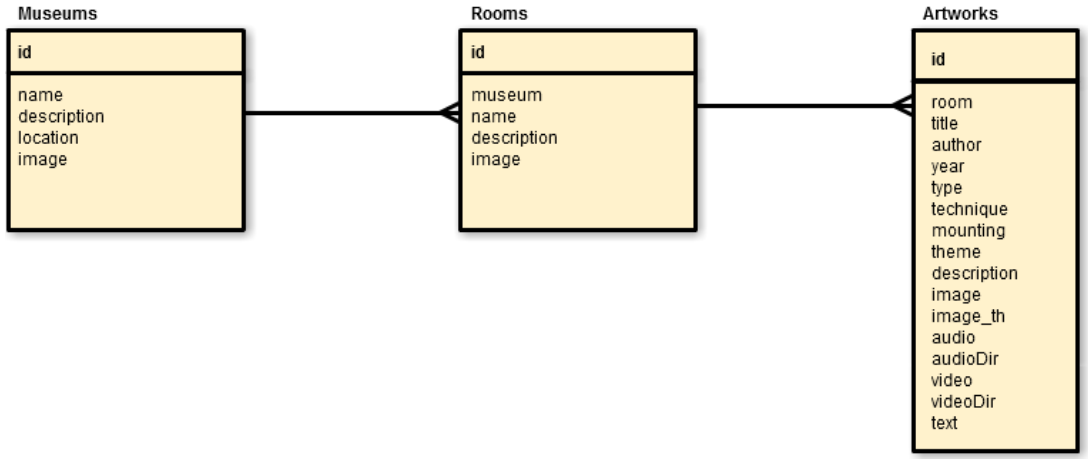

Gráfico 3.5. Tablas y relaciones que forman el modelo de datos de la aplicación

3.2.2 Comunicaciones

Se ha usado como protocolo de comunicaciones JSON, por su simplicidad, ser el más estandarizado actualmente y cubrir las necesidades de la aplicación. Como no se implementa lado del servidor, la respuesta JSON que éste proporcionaría viene definida en ficheros de texto con extensión *.json*.

3.2.3 Servidor

Aunque las peticiones que puede realizar el cliente son bastante simples y limitadas, el servidor debe proporcionarles respuesta. Estas son las peticiones disponibles, la respuesta, y el fichero *.json* que la modela.

Petición cliente	Respuesta servidor	Fichero .json
lista_museos	listado de museos	museums.json
lista_salas(museo_id)	listado de salas del museo X	rooms(museo_id).json
lista_obras(sala_id)	listado de obras de la sala X	artworks(sala_id).json

4. Desarrollo de la aplicación

4.1 Cliente

Como se comentó en el apartado *1.4 Tecnologías a utilizar*, para el desarrollo de la aplicación cliente se ha usado el *Samsung SDK 4.5*, compatible con TVs *Samsung* fabricadas a partir de 2013. Según la *Tabla 1.1* del mismo apartado, estos modelos son compatibles con *HTML 5*, *CSS 3* y *Javascript V8*. Aunque modelos fabricados en años anteriores (2011 y 2012) también admiten el uso de *HTML 5* y *CSS 3*, no se garantiza que la aplicación sea 100% compatible.

El entorno de desarrollo consiste en un entorno *Eclipse* con el *Samsung Smart TV SDK* integrado. Permite crear dos tipos de proyectos:

- Proyecto básico: basado en *javascript*, proporciona un *framework* para el desarrollo de UI (User´s Interface, Interfaz de Usuario) que provee de envolturas de alto nivel que facilitan el desarrollo de la aplicación. Se basa en el concepto de *Escenas*.
- Proyecto *javascript*: Plantilla básica para escribir las aplicaciones en *javascript* puro. Proporciona mayor control a costa de una mayor complejidad. **Es el que se ha usado en este desarrollo**.

Existen a disposición del programador unas APIs de *Samsumg* (en *javascript*), agrupadas en distintas librerías que proporcionan distintas características que facilitan el desarrollo de la aplicación. Para evitar tener que hacer grandes modificaciones de este desarrollo a la hora de hacerlo compatible con otro tipo de TVs, se ha hecho un uso limitado de esta API.

4.1.1 Estructura de la aplicación

La estructura básica de un proyecto *javascript* consiste en un archivo *config.xml*, que contiene los parámetros de configuración de la aplicación, y un archivo *index.html*, que constituye el punto de acceso a la aplicación.

```
<?xml version="1.0" encoding="UTF-8"?>
<widget>
    <cpname itemtype="string"></cpname>
    <cplogo itemtype="string"></cplogo>
    <cpauthjs itemtype="string"></cpauthjs>
    <ThumbIcon itemtype="string">icon/sampleIcon_106_87.png</ThumbIcon>
    <BigThumbIcon itemtype="string">icon/sampleIcon_115_95.png</BigThumbIcon>
    <ListIcon itemtype="string">icon/sampleIcon_85_70.png</ListIcon>
    <BigListIcon itemtype="string">icon/sampleIcon_95_78.png</BigListIcon>
    <category itemtype="string"></category>
    <autoUpdate itemtype="boolean">n</autoUpdate>
    <ver itemtype="string">0.100</ver>
    <mgrver itemtype="string"></mgrver>
    <fullwidget itemtype="boolean">y</fullwidget>
    <type itemtype="string">user</type>
    <srcctl itemtype="boolean">y</srcctl>
    <ticker itemtype="boolean">n</ticker>
    <childlock itemtype="boolean">n</childlock>
    <videomute itemtype="boolean">n</videomute>
    <dcont itemtype="boolean">y</dcont>
    <widgetname itemtype="string">Museum</widgetname>
    <description itemtype="string"></description>
    <width itemtype="string">1280</width>
    <height itemtype="string">1024</height>
    <author itemtype="group">
        <name itemtype="string"></name>
        <email itemtype="string"></email>
        <link itemtype="string"></link>
        <organization itemtype="string"></organization>
    </author>
</widget>
```

Gráfico 4.1. Archivo config.xml

A estos archivos, se añadirán los archivos auxiliares necesarios, ya sean de diseño (con extensión *.css*), imágenes usadas en el diseño de la aplicación, o conteniendo lógica de la aplicación o librerías externas como *jQuery* (con extensión *.js*). Estos archivos se organizan en distintas carpetas para facilitar su localización. Los **archivos conteniendo los datos finales que se muestran al usuario**, (como datos de los museos, fotografías de cuadros, vídeos explicativos, etc.) **no forman parte de la estructura de la aplicación, sino que se recuperan desde un repositorio externo**, garantizando así una independencia de la lógica de la aplicación respecto a los datos. Generalmente, hará falta un *Main.js* y un *Main.css*, aunque pueden crearse los *js* y *css* que sean necesarios. Así, la estructura organizativa de la aplicación queda de la siguiente forma:

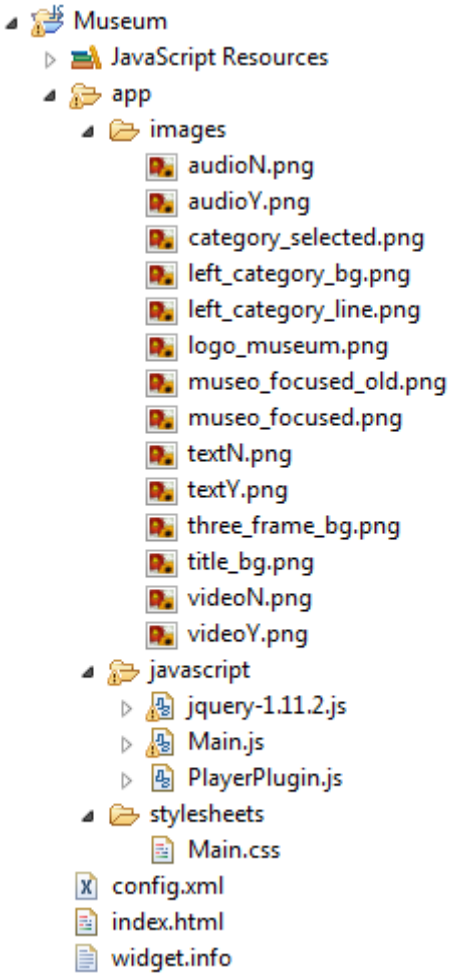

Gráfico 4.2. Estructura de ficheros de la aplicación

4.1.2 Elementos de la API Samsumg utilizados

sfKeyHelp

Muestra la barra de ayuda. Cuando se crea, se inicializa con los valores de los botones del control remoto que va a mostrar, por lo que es necesario crear tantas como barras distintas se quieran mostrar. En esta aplicación se usan cuatro barras distintas. Una instanciación de una de ellas, la que es visible cuando se está ejecutando un vídeo o audio, como ejemplo:

```
$('#keyHelpPlayer').sfKeyHelp({
	'iconset' : 'WHITE',
	'play'    : 'Play',
	'stop'    : 'Stop',
	'pause'   : 'Pausa/Continuar',
	'rewff'   : 'Atrasar/Adelantar',
	'return'  : 'Volver'
});
```

sfKeyScroll

Muestra una barra de scroll, inicializada con un número total de páginas. Provee el elemento visual en sí, aunque la funcionalidad hay que implementarla en su mayor parte. Es necesario calcular el número de páginas en función del número total de elementos a mostrar, y en función del elemento que vaya obteniendo el foco, realizar la paginación de forma manual, asignando la página actual correspondiente a la barra de scroll para recalcular su posición (la altura parcial de la barra dentro de su altura total) y cambiando la página que se muestra si fuera necesario.

```
$('#MainScroll').sfScroll({
	pages: totalPages
});
```

Otros elementos necesarios

Para el correcto funcionamiento de la aplicación además es necesario usar dos elementos presentes en la API que son esenciales.

El primero de ellos es *Manejador de Aplicaciones* (*Application Manager*). Es el encargado de lanzar la aplicación. Para invocarlo, es necesario cargar el archivo *.js* correspondiente en el *index.html* y llamarlo cuando la página se carga en el *Main.onLoad().*

El segundo es el listener/manejador del evento de tecla pulsada (*keyDown*) en el control remoto. Detecta cuando se pulsa una tecla en el control remoto y realiza la acción correspondiente. El comportamiento de la aplicación viene definido en gran medida por este manejador.

La integración de ambos elementos sería, primero en el *index.html*:

```
<html>
    <head>
        ...
        <script type="text/javascript" language="javascript"
                src="$MANAGER_WIDGET/Common/API/Widget.js"></script>
        <script type="text/javascript" language="javascript"
                src="$MANAGER_WIDGET/Common/API/TVKeyValue.js"></script>

        <script type="text/javascript" language="javascript"
                src="app/javascript/Main.js"></script>
        ...
    </head>

    <body onload="Main.onLoad();" onunload="Main.onUnload();">
        <!-- Ancla para eventos de tecla en control remoto -->
        <a href="javascript:void(0);" id="anchor" onkeydown="Main.keyDown();"></a>
        ...
    </body>
</html>
```

Gráfico 4.3. Elementos necesarios en el fichero *index.html*

Y en el *Main.js*:

```
var widgetAPI = new Common.API.Widget();
var tvKey = new Common.API.TVKeyValue();
var Main = {};
Main.onLoad = function() {
    // Habilita el procesamiento de eventos de teclas
    this.enableKeys();
// Lanza el evento de aplicación preparada
    widgetAPI.sendReadyEvent();
}
Main.enableKeys = function(){
    document.getElementById("anchor").focus();
};
Main.keyDown = function(){
    // Procesa el evento de tecla pulsada
    var keyCode = event.keyCode;

    switch(keyCode)
    {
        case tvKey.KEY_RETURN:
        case tvKey.KEY_LEFT:
        case tvKey.KEY_RIGHT:
        …
    }
};
```

Gráfico 4.4. Elementos necesarios en el fichero *Main.js*

Como se ha comentado, gran parte de la lógica de la aplicación está presente en el manejador del evento de tecla pulsada, ya que el flujo de la aplicación depende mayormente de la entrada del usuario a través del control remoto. Merece la pena estudiar en detalle este manejador.

4.1.3 El manejador del control remoto

El manejador de evento de tecla pulsada en el control remoto debe generar las acciones y respuestas adecuadas en función de la tecla pulsada en el control remoto. Esto implica llevar un control del flujo de la aplicación, ya que puede ser que una tecla tenga funcionalidad o no según el nivel de navegación en el que se encuentre el usuario dentro la aplicación.

Por ello, se hace necesario llevar un registro de en qué nivel está el usuario en cada momento, para así aplicar la respuesta adecuada. Esto se hace con la variable *breadcrumbLevel*. En el siguiente ejemplo de código, se muestra un trozo del manejador donde se detalla el comportamiento si se pulsa el botón *ENTER* y el botón *RED* (botón rojo). Se observa que, dependiendo del nivel de navegación, se realizarán unas acciones determinadas u otras (o ninguna).

```
Main.keyDown = function(){

    var keyCode = event.keyCode;

    switch(keyCode)
    {
        case tvKey.KEY_ENTER:
        case tvKey.KEY_PANEL_ENTER:
            switch (breadcrumbLevel) {
                case 1: // Nivel museos, entrando a salas
                    selectedRoom = 0;
                    totalRooms = 0;
                    showRooms();
                    focus("s0");
                    breadcrumbLevelUp(museums.museums[selectedMuseum].name);
                    break;
                case 2: // Nivel salas, entrando a obras
                    showArtworks();
                    document.getElementById("content_list").style.display = "none";
                    hideScrollBar();
                    document.getElementById("wrapper_bu").style.display = "block";
                    breadcrumbLevelUp(rooms[selectedMuseum].rooms[selectedRoom].name);
                    $('#keyHelpMain').sfKeyHelp('hide');
                    $('#keyHelpGallery').sfKeyHelp('show');
                    break;
                case 3: // Nivel obras, entrando a pantalla completa
                    Gallery.enterFullScreen();
                    break;
            }
            break;
        case tvKey.KEY_RED:
            alert("RED");
            switch (breadcrumbLevel) {
                case 4: // Nivel obra pantalla completa
                    // Mostrar/ocultar descripción
                    $("#imageDesc").toggle(1000);
                    break;
            }
            break;
        default:
            alert("Unhandled key");
            break;
    }
};
```

Gráfico 4.5. Detalle parcial del manejador del evento de tecla pulsada en el control remoto

4.1.3 Animaciones

Para dar un aspecto visual más atractivo, se han aprovechado las capacidades que ofrece la librería *jQuery* y se han aplicado animaciones en ciertas transiciones. Estas se dan en dos casos principales

- Scroll: Cuando la navegación por los listados implica un salto de página, éste se hace con una animación. Se muestra la nueva página y el foco pasa al primer elemento del listado.
- Capas: Cuando se muestra la obra a pantalla completa, se pueden pulsar los botones rojo, verde o amarillo para mostrar información adicional en forma de texto, vídeo o audio. Esta información se muestra en una nueva capa traslúcida que se hace visible (o invisible, al cerrase) mediante otra transición animada.

4.1.4 Galería 3D

Quizás el elemento más espectacular visualmente sea la galería con aspecto 3D que se muestra al entrar en una sala. Este elemento existía previamente pero ha sido adaptado específicamente para ser utilizado en este desarrollo.

4.1.5 Reproductor multimedia

Existen varias formas de reproducir contenido multimedia en una *Smart TV*, aunque la mayoría de ellas hacen uso de *plugins* que proporcionan distintas APIs (*webapi, deviceapi*). Se ha optado por usar un método lo menos intrusivo posible, y para ello se ha utilizado la etiqueta <*object*> de *html* e invocando a un plugin de *Samsung*, de la siguiente forma:

```
<object id="videoPlugin" classid="clsid:SAMSUNG-INFOLINK-SEF"></object>
```

Y se ha creado un archivo *PlayerPlugin.js* para su manejo específico. En él se implementan las operaciones de *play, stop, pause, resume, ff,* y *rw*. Cada vez que se solicita la ejecución (*play*), se genera un nuevo reproductor. Esta invocación se hace de forma genérica sobre un elemento del DOM llamado *el*, de forma que se pueden usar varios *plugins* simultáneamente definidos sobre distintos elementos. Así, además del reproductor de vídeo, es posible definir otro *plugin* para reproducir los archivos de sólo audio. Este último reproductor, como no necesita ser visionado, se sitúa fuera del área de visión de contenidos.

```
var url;
var el;
var plugin;
var isPlaying = false;
var isPaused = false;

function play(url, element) {

    el = element;
    plugin = document.getElementById(el);
    plugin.Open('Player', '1.112', 'Player');
    plugin.Execute("InitPlayer", url);
    plugin.Execute("SetDisplayArea", 75, 50, 570, 420);
    plugin.Execute("SetInitialBufferSize", 400*1024);
    plugin.Execute("StartPlayback");
    isPlaying = true;
}

function stop() {
    if (isPlaying) {
        isPlaying = false;
        isPaused = false;
        plugin.Execute("Stop");
    }
}

function pause() {
    if (isPlaying && !isPaused) {
        isPaused = true;
        plugin.Execute("Pause");
    }
    else if (isPlaying && isPaused) {
        resume();
    }
}

function resume() {
    isPaused = false;
    plugin.Execute("Resume");
}

function ff() {
    plugin.Execute("JumpForward", 10);
}

function rw() {
    plugin.Execute("JumpBackward", 10);
}
```

Gráfico 4.6. ***PlayerPlugin.js***

4.1.6 Petición AJAX

Para solicitar datos externos, el cliente realiza una petición AJAX de la siguiente forma:

```
$.ajax({
    url: "url",
    data: "nocache=" + Math.random(),
    type: "GET",
    dataType: "json",
    async: false,
    success: function(source){
        data = source;
        ...
    },
    error: function(dato){
        alert("ERROR!");
    }
});
```

Gráfico 4.7. Llamada AJAX del lado del cliente

Donde el campo *"url"* será la dirección donde se encuentran los datos que se requieren (o el servicio que los proporciona), en formato JSON. En caso de éxito, los datos recibidos quedan formateados para su tratamiento en la variable *data*.

4.1.7 Almacenamiento de datos

Con el fin de no realizar comunicaciones redundantes, los datos obtenidos en las peticiones que el cliente va realizando son almacenados, de forma que si se solicitan datos que ya se pidieron con anterioridad, se recuperan de la estructura de datos local y no mediante una nueva comunicación con el servidor.

Para tal fin, se ha creado un array tridimensional donde se modelan los museos (1ª dimensión), las salas de cada museo (2ª dimensión) y las obras de cada sala (3ª dimensión). Los archivos con información multimedia (vídeos, audios y fotos) no son almacenados, por lo que sí son requeridos al repositorio multimedia en cada petición (apartado 4.3.1).

4.2 Servidor

Como se ha comentado en apartados anteriores, la implementación del lado del servidor queda fuera del alcance de este TFG. Para el correcto funcionamiento de la aplicación, se ha simulado una comunicación y respuesta del servidor, usando ficheros *.json* que contienen la respuesta que enviaría el servidor. Los datos contenidos en estos archivos dependen de la petición que efectúa el cliente, que pueden ser tres:

4.2.1 Solicitar datos de museos

Es la petición principal, ya que se realiza al iniciar la aplicación. Lo primero que se debe mostrar es el listado de museos disponibles. El formato de la respuesta recibida tiene el siguiente formato:

Consulta: get_museums()

Ejemplo de respuesta JSON:


```
{
 "museums":[
 {
"id" : "3",
"name" : "Museo del Prado",
"description" : "El museo más importante del mundo en pintura europea",
"location" : "Madrid",
"image" : "http://192.168.1.128/media/museum3/prado.jpg"
 },
 {
"id" : "2",
"name" : "Museo Vaticano",
"description" : "Conjunto de galerías y demás estancias de valor artístico
"location" : "Roma",
"image" : "http://192.168.1.128/media/museum2/vaticano.jpeg"
 },
 {
 "id" : "1",
 "name" : "Museo del Louvre",
 "description" : "Consagrado al arte anterior al impresionismo, tanto bella
 "location" : "París",
 "image" : "http://192.168.1.128/media/museum1/louvre.jpg"
 },
 {
 "id" : "4",
 "name" : "National Gallery",
 "description" : "Pinacoteca con más de 2300 obras, mayormente europeas, de
 "location" : "Londres",
 "image" : "http://192.168.1.128/media/museum4/national_gallery.jpg"
 }
 ]
}
```

4.2.4 Solicitar datos de salas

Esta petición se realiza cuando el usuario entra en un museo. Se solicita el listado de salas que forman ese museo:

Consulta: get_rooms(museum_id)

Ejemplo de respuesta (parcial) JSON:


```
{
 "rooms":[
 {
 "id" : "1",
 "name" : "Diego de Velázquez",
 "museum" : "3",
 "description" : "",
 "image" : "http://192.168.1.128/media/museum3/room1/prado-room1.jpg"
 },
 {
"id" : "2",
"name" : "Francisco de Goya",
"museum" : "3",
"description" : "",
"image" : "http://192.168.1.128/media/museum3/room2/prado-room2.jpg"
 },
 {
"id" : "3",
"name" : "Pinturas",
"museum" : "3",
"description" : "",
"image" : "http://192.168.1.128/media/museum3/room3/prado-room3.jpg"
 },
{
"id" : "4",
"name" : "Esculturas",
```


4.2.3 Solicitar datos de obras

Esta petición se realiza cuando el usuario entra en una sala. Se solicita el listado de obras que forman esa sala:

Consulta: get_artworks(room_id)

Ejemplo de respuesta (parcial) JSON:


```
{
 "artworks":[
{
 "id" : "15",
 "title" : "El bufón don Sebastián de Morra",
 "author" : "Diego Rodríguez de Silva y Velázquez",
 "year" : "1649",
 "type" : "pintura",
 "technique" : "óleo",
 "mounting" : "lienzo",
 "theme" : "retrato",
 "school" : "española",
 "description" : "Velázquez juega con el contraste entre la expresión seria y reflexiva del enanc
 reflexionar sobre la condición humana",
 "room" : "1",
 "museum" : "3",
 "image" : "http://192.168.1.128/media/museum3/room1/Velazquez-BufonDonSebastianDeMorra.jpg",
 "image_th" : "http://192.168.1.128/media/museum3/room1/Velazquez-BufonDonSebastianDeMorra.jpg",
 "audio" : "Y",
 "video" : "N",
 "text" : "Y",
 "audioDir" : "http://192.168.1.128/media/museum3/room1/elbufondonsebastiandemorra.mp3",
 "videoDir" : "" },
{
 "id" : "16",
 "title" : "La Coronación de la virgen",
 "author" : "Diego Rodríguez de Silva y Velázquez",
 "year" : "1636",
 "type" : "pintura",
 "technique" : "óleo",
 "mounting" : "lienzo",
 "theme" : "religión",
 "school" : "española",
 "description" : "Cuadro de devoción privada de Isabel de Borbón, esposa de Felipe IV, en el que
 meditación íntima",
 "room" : "1",
 "museum" : "3",
 "image" : "http://192.168.1.128/media/museum3/room1/Velazquez-CoronacionDeLaVirgen.jpg",
 "image_th" : "http://192.168.1.128/media/museum3/room1/Velazquez-CoronacionDeLaVirgen.jpg",
```


4.3 Los datos

Podemos diferenciar entre **dos tipos de datos**: los datos que forman la estructura de la aplicación en sí (listados de museos, salas y obras) y los de formato multimedia (archivos de vídeo, audio y foto). Dada su distinta naturaleza, también será distinto su almacenamiento.

Al igual que en el lado del servidor, la implementación del modelo de datos queda fuera del alcance de este TFG, aunque se realiza una propuesta.

4.3.1 Almacén de datos

El almacén de datos estará basado en un *SGBD* (Sistema Gestor de Base de Datos) y contendrá los datos relevantes a la estructura de la aplicación. Contendrá las tablas necesarias para realizar un modelado de los museos, salas y obras disponibles, cada uno de ellos con sus atributos correspondientes (según el modelo de datos del apartado 3.2.1). Debe ser accesible mediante una aplicación del lado del servidor (servicio web o similar).

4.3.2 Repositorio multimedia

El repositorio multimedia consiste en un sistema de ficheros accesible mediante una *URL* (alojado en un servidor web, por ejemplo). La organización del sistema de ficheros no es fija y se puede estructurar como mejor se considere. El único requisito es la concordancia con el valor de los atributos que identifican estos recursos en la base de datos, por ejemplo, si el atributo *"image"* de una obra identifica la *url* de la fotografía de esta obra, ésta *url* debe ser accesible y contener realmente dicha fotografía.

5. Pruebas y validación

Para poder probar la aplicación, ya sea en el emulador de TV o en una TV real, es necesario disponer de un servidor web donde estarán alojados tanto el almacén de datos (apartado 4.3.1) como el repositorio multimedia (apartado 4.3.2). Los datos del almacén deben ser congruentes, es decir, las referencias a *urls* de ficheros deben corresponderse con las *urls* del repositorio de datos.

5.1 Prueba en emulador

Como se comentó en el apartado 1.4, el emulador de TV es en realidad una máquina virtual que se ejecuta dentro del virtualizador *VirtualBox* de *Oracle*. Por tanto, se debe disponer de este software instalado (se recomienda la versión 4.2.24), y un archivo con extensión *.ova* con la imagen del emulador (*2013 Samsung Smart TV Emulator.ova*), cargado en el virtualizador.

Cuando se produce la carga de la imagen del emulador en el virtualizador se muestran distintas opciones de configuración. La principal es la de *"Carpeta Compartida",* donde se define una carpeta con ruta por defecto *C:\share\Apps*. Esta ruta se puede modificar, pero lo importante es que en esta carpeta se debe alojar nuestra aplicación.

Una vez configurada correctamente, se puede iniciar la máquina virtual. Tras un proceso de arranque, mostrará la imagen de una pantalla de *Samsung Smart TV*, junto a un dispositivo controlador (mando de control remoto), que nos servirá para interactuar con la aplicación. En la parte superior existen unos botones funcionales. El botón *Open* nos permite seleccionar una aplicación de la carpeta compartida para ejecutarla. Si nuestra aplicación se encuentra ahí, aparecerá para ser seleccionada y ejecutada, dando comienzo a la emulación.

Debido a diferencias evidentes con la TV real y a que puede haber características que no estén disponibles en el emulador, es posible que el comportamiento de la aplicación difiera algo con la TV real.

5.2 Prueba en TV

El entorno real de TV difiere al del emulador en que se dispone de menos memoria física, por lo que pueden ocurrir errores por falta de memoria. También, la respuesta del control remoto puede ser diferente, al tener un tiempo de respuesta distinto al pulsar las teclas. En cambio, los gráficos

pueden ser mejor evaluados al tener una respuesta de la aplicación en un entorno real.

Una aplicación subida directamente desde el PC a la TV se conoce como Aplicación de Usuario. Para subir nuestra Aplicación de Usuario a la TV debemos disponer de un servidor web (Apache, por ejemplo), donde debemos alojar nuestra aplicación, correctamente empaquetada.

Entonces, los pasos necesarios para probar nuestra Aplicación de Usuario en la TV serían los siguientes:

En el PC:

- Editar el archivo *config.xml*, e incluir `<type>user</type>` entre las etiquetas `<widget></widget>`.
- Hacer click en la opción del SDK *Empaquetar aplicación (App packaging).*
- Introducir:
  - Nombre de la aplicación y el número de versión.
  - Región donde será usada la aplicación.
  - Fecha de empaquetamiento.
- Seleccionar *Actualizar los archivos empaquetados en el servidor (Update the packaged files on the server)* y pulsar *OK.*
- Transferir al servidor el archivo `widgetlist.xml` y la carpeta `widget` y su contenido, de forma que sean accesibles.

Tras esto, nuestra aplicación estaría correctamente empaquetada y alojada en el servidor web, esperando a ser descargada en nuestra TV.

En la TV:

- Configurar los datos de entorno de red en la TV.
- Crear la cuenta de usuario *"develop"* si no existía (en TV anteriores a 2013) y loguearse con ella (dejar contraseña en blanco).
- Introducir la IP del PC de desarrollo (donde está el servidor web). Puede ser necesario reiniciar la TV.
- Sincronizar Aplicaciones de Usuario (descargar aplicaciones desde el servidor web).

Después de esto, la aplicación estaría disponible para su prueba y ejecución en el *Smart Hub*. Si fuera necesario realizar algún cambio en la aplicación, ésta se debería empaquetar de nuevo, alojar en el servidor web y volver a descargar e instalar en la TV para probar la aplicación con los últimos cambios efectuados.

5.3 Validando y lanzando la aplicación al mercado de *Samsung*

Después de que la aplicación haya sido depurada y testeada localmente, tanto en el emulador como en una TV real, está lista para ser lanzada al mercado de *Samsung*, conocido como *Samsung Apps*. Durante este proceso de lanzamiento, la aplicación pasará por nuevas pruebas realizadas por el equipo de *Samsung*. Si las supera, quedará aprobada para su publicación en el *Samsung Apps,* desde donde los usuarios la podrán descargar a sus TVs.

6. Conclusiones

El que casi todos los hogares cuenten con una o varias TVs (que tarde o temprano todas acabarán siendo TV con gran área de visionado conectadas a Internet) y ésta se encuentre normalmente en el núcleo del hogar, la hacen el dispositivo con más futuro de cara a los desarrolladores, aunque estos se han encontrado con la falta de consenso inicial entre fabricantes a la hora de desarrollar su producto bajo un estándar uniforme. Quizás por ello el mercado de aplicaciones para TVs no esté tan saturado como el de otros dispositivos. Y como consecuencia de ello, la información disponible para un desarrollador sea bastante limitada.

Aunque en los últimos tiempos parece que los fabricantes están convergiendo a un estándar común, primero *HTML 5* y *CSS 3*, sin uso de APIs propias, y luego *Android*, lo que sin duda va a facilitar y propiciar una enorme expansión de contenidos disponibles para TV inteligentes.

En lo que respecta a la aplicación aquí desarrollada, se ha evitado hacer uso en la medida de lo posible de elementos propietarios que se salieran del estándar, para hacerla lo más compatible posible con la mayoría de TVs en el mercado. Se ha obtenido una aplicación válida para TV *Samsung* fabricadas desde 2013 en adelante, y compatible con otras muchas de similares características (conectables a Internet y con centro de aplicaciones disponible) con mínimos cambios.

En este TFG se dejan abiertas distintas líneas de desarrollo. Se puede implementar el lado del servidor (del que únicamente se ha definido su arquitectura). Además, se podría implementar una aplicación gestora, vía web por ejemplo, que permitiera acceder al almacén de datos para modificar o añadir nuevos contenidos.

También se podría utilizar el concepto de la aplicación, el de repositorio de contenido multimedia, para implementar otras aplicaciones basadas en este concepto. Por ejemplo, un repositorio de contenido por usuario, donde cada usuario identificado por unas credenciales, podría albergar el contenido de audio y vídeo propio que estimase, y que previa identificación del usuario, pudiera ser recuperado y visionado en cualquier TV. Gran parte de la lógica existente en este TFG sería aprovechable para este posible desarrollo.

En cualquier caso, lo que sí parece es que las TVs no han sufrido todavía el colapso de aplicaciones que sí sufren otros dispositivos (los teléfonos móviles), por lo que puede resultar un mercado interesante para futuros proyectos de desarrollo software.

7. Referencias bibliográficas

· Foro de desarrolladores para Smart TV de Samsung.
http://www.samsungdforum.com/

Referencias a la API de desarrollo de aplicaciones Smart TV, guía de estilo, foro de desarrolladores, etc.

· Foro de desarrolladores para Smart TV de Samsung en español
http://samsungstad.com/

· Mercado de aplicaciones Samsung para Smart TV (España).
http://www.samsung.com/es/tv-apps/

· Mercado de aplicaciones Samsung para Smart TV (UK).
http://www.samsung.com/uk/tvapps/

· Registro de la aplicación en el market de Samsung Smart TV.
http://seller.samsungapps.com

8. Anexos técnicos

8.1 Manual de usuario

Debido al dispositivo al que se orienta este desarrollo y a la gran variedad del público objetivo, se ha intentado hacer una aplicación con un uso sencillo e intuitivo. Aun así, se detalla este pequeño manual que pormenoriza toda la funcionalidad disponible.

La aplicación, como se comenta en apartados anteriores, se estructura en niveles de navegación. En común a todos los niveles, existe una barra de ayuda, siempre visible en la parte inferior, que detalla qué botones del control remoto tienen funcionalidad (básicamente, flechas de navegación, colores o controles de reproducción). Además, en todos los niveles, con el botón *Enter* se avanza al nivel inferior y con el botón *Return* se vuelve al nivel superior. Si se estuviera ya en el nivel más alto, se saldría de la aplicación.

La barra de navegación indica en qué nivel se encuentra el usuario.

8.1.1 Nivel 1: Museos

Este es el nivel inicial, donde arranca la aplicación. Muestra el listado de museos disponibles. La barra de ayuda muestra los botones *Arriba* y *Abajo*, para navegar por el listado y *Enter*, para entrar en el museo seleccionado. Con *Return*, al estar en el nivel más superior, se cerraría la aplicación.

Gráfico 8.1. Nivel 1, listado de museos

8.1.2 Nivel 2: Salas

Este nivel presenta a aspecto idéntico al nivel 1, pero referido a un listado de salas. Por lo tanto, la barra de ayuda muestrea los mismos botones y las operaciones disponibles son análogas, con la diferencia que pulsando *Enter* se entraría al listado de obras de la sala seleccionada y con *Return* se volvería al nivel superior de listado de museos.

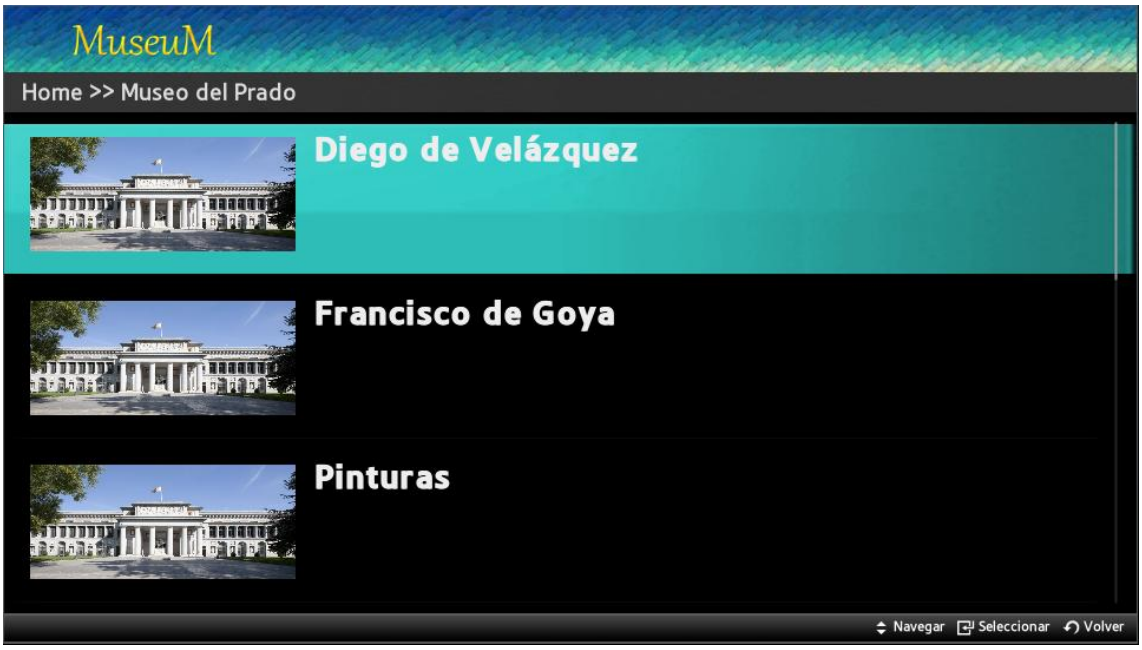

Gráfico 8.2. Nivel 2, listado de salas

8.1.1 Nivel 3: Obras

Aquí la disposición del listado varía ligeramente, por lo que la navegación entre las distintas obras se hace de forma lateral, mediante los botones *Derecha* e *Izquierda*. Con *Enter* se pasaría a ver la obra en detalle a pantalla completa, y con *Return* se volvería al listado de salas. La barra de ayuda se modifica para mostrar dicha información.

Gráfico 8.3. Nivel 3, listado de obras de una sala

8.1.4 Nivel 4: Detalle de las obras

En este nivel se muestra la obra a pantalla completa, y se posibilita el ver toda la información adicional disponible (vídeo, audio o texto descriptivo) de la obra. Además, se permite el seguir navegando por las obras sin necesidad de subir a la galería de obras del nivel superior. Pulsando flecha *Derecha* o *Izquierda*, se actualiza la información con la de la obra correspondiente.

Gráfico 8.4. Nivel 4, detalle de una obra a pantalla completa

Con los botones de colores se consigue el acceso a la información adicional. Con el botón *Rojo*, se despliega el panel que contiene el texto descriptivo. Consiste en unos datos generales sobre la obra y una pequeña descripción. También se muestran unos iconos que indican si la obra dispone de video o audio adicional. Si se vuelve a pulsar, se oculta el panel.

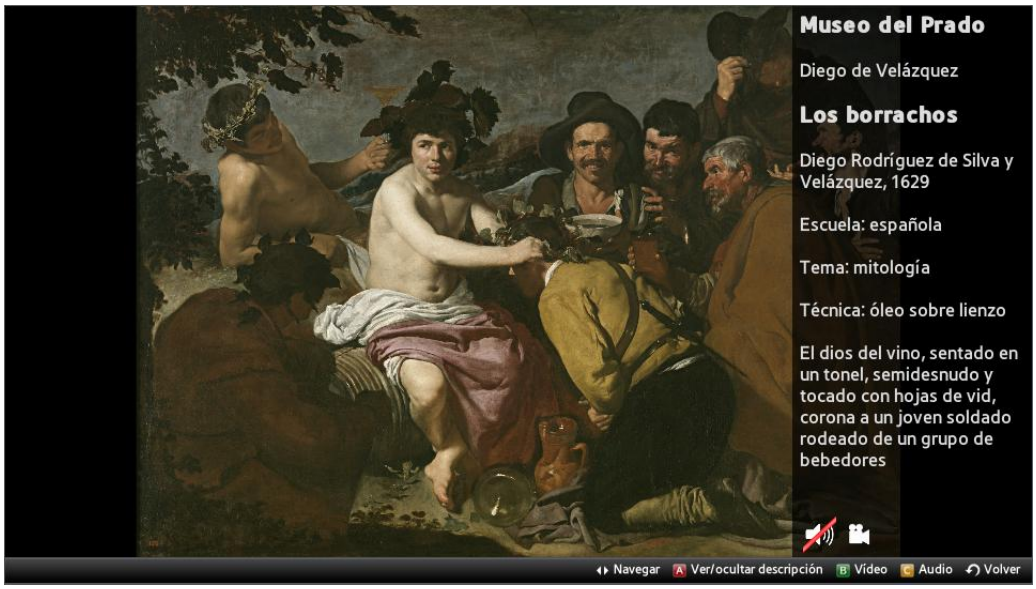

Gráfico 8.5. Botón rojo pulsado, mostrando texto descriptivo

Con el botón *Verde* se despliega el panel de reproducción de vídeo. Si hay vídeo disponible, automáticamente empezará su ejecución y la barra de ayuda se actualizará con la información de los botones de control de reproducción. Si no hay vídeo, el panel desplegado muestra un mensaje indicando tal circunstancia. Volviendo a pulsar el botón *Verde*, se pliega el panel.

Gráfico 8.6. Botón verde pulsado, reproduciendo vídeo. Se mantiene el panel del texto descriptivo mostrado con el botón rojo pulsado con anterioridad.

El botón *Amarillo* para la reproducción del audio tiene un funcionamiento análogo al botón *Verde* de vídeo, con la salvedad de que el audio no necesita (ni puede) ser mostrado, por lo que no se despliega ningún panel adicional. Este panel aparece únicamente en el caso de que no haya un audio disponible para su reproducción, para mostrar el mensaje de error correspondiente. El audio también dispone de controles de reproducción, por lo que si hay archivo de audio disponible y éste comienza a reproducirse, la barra de ayuda se actualiza también mostrando los botones de los controles de reproducción.

Gráfico 8.7. Botón amarillo pulsado y controles de reproducción mostrándose en la barra de ayuda. El texto descriptivo permanece visible mientras no se oculte explícitamente

Los controles de reproducción disponibles son:

- *Play*: Inicia la reproducción desde el inicio.
- *Stop*: Para la reproducción.
- *Pause*: Pausa una reproducción en curso, o la reanuda si estaba pausada.
- *Avance rápido*: avanza el vídeo 10 segundos (reproduciéndose o en pausa).
- *Retroceso rápido*: retrocede el vídeo 10 segundos (reproduciéndose o en pausa).

Como se ha comentado, la barra de ayuda sólo muestra estos controles cuando hay contenido multimedia disponible para su reproducción.

El panel de texto descriptivo tiene persistencia de estado, es decir, podemos navegar por las distintas obras y el panel no se ocultará hasta que se indique expresamente. Esto ocurre incluso si se sube un nivel de navegación, hasta la galería de obras. Si entramos a otra obra y el texto descriptivo se dejó desplegado anteriormente, se verá de nuevo desplegado.

No ocurre así con los paneles de vídeo y audio. Si están desplegados y se navega a otra obra, la reproducción se interrumpe y el panel se pliega, quedando oculto.

Pulsando repetidamente el botón *Return*, se irá subiendo de nivel hasta llegar al nivel superior y salir de la aplicación.